IN THE CLAIMS

None of the claims are amended or canceled. Claim 1-42 are repeated herein only for reviewing convenience.

1. (Original) A method comprising:

initiating a power increase for a potentially needed functional unit to an operable power level, if the potentially needed functional unit has a present power level that is lower than the operable power level, wherein the potentially needed functional unit is identified based on a determination of whether the potentially needed functional unit is operable to execute at least one software instruction stored within an instruction cache.

2. (Original) The method of claim 1, further comprising:

fetching one or more software instructions into the instruction cache.

3. (Original) The method of claim 2, wherein fetching the one or more instructions comprises fetching the one or more instructions into a conventional cache.

4. (Original) The method of claim 2, wherein fetching the one or more instructions comprises fetching the one or more instructions into a trace cache.

5. (Original) The method of claim 1, further comprising:

fetching a line of one or more software instructions into the instruction cache;

generating and storing an information vector for the line, wherein the information vector identifies a set of functional units that are operable to execute the one or more software instructions; and

identifying the potentially needed functional unit based on the information vector.

6. (Original) The method of claim 1, further comprising:

indicating power status information for a set of functional units, wherein the power status information indicates whether a functional unit, within the set of functional units, has a present power level that places the functional unit in an operable power state or a low power state.

7. (Original) The method of claim 1, further comprising:

incrementing a use counter for a functional unit when a software instruction is fetched into the instruction cache, and when the functional unit is operable to execute at least part of the software instruction.

8. (Original) The method of claim 7, further comprising:

decrementing the use counter for the functional unit when the software instruction is eliminated from the instruction cache.

9. (Original) The method of claim 1, further comprising:

selecting one or more selected lines of software instructions stored within the cache; and

identifying the potentially needed functional unit as a functional unit that is operable to execute at least one software instruction stored within the one or more selected lines.

10. (Original) The method of claim 1, further comprising:

activating a line of software instructions stored within the cache; and

identifying the potentially needed functional unit as a functional unit that is operable to execute at least one software instruction stored within the line.

11. (Original) The method of claim 1, further comprising:

identifying an unneeded functional unit as a functional unit that is not operable to execute the at least one software instruction; and

initiating a power decrease for the unneeded functional unit, if the unneeded functional unit has a second present power level that is greater than or equal to a second operable power level.

12. (Original) The method of claim 11, wherein initiating the power decrease comprises:

initiating the power decrease for the unneeded functional unit after execution is complete of any in-flight instructions that use the unneeded functional unit.

13. (Original) The method of claim 1, wherein initiating the power increase comprises:

initiating the power increase for a functional unit selected from a group of functional units that includes one or more floating-point units, multipliers, dividers, shifters, digital signal processors, co-processors, application specific integrated circuits, data processing engines, debug logic blocks, encryption units and key-generation units.

14. (Original) The method of claim 1, wherein initiating the power increase comprises:

determining a selected operable power level from one of multiple operable power levels, wherein the selected operable power level is selected based on an expected result latency; and

initiating the power increase to the selected operable power level.

15. (Original) A method comprising:

fetching one or more lines of software instructions into an instruction cache, which is accessible to a processing engine;

identifying potentially needed functional units as functional units that are operable to execute at least one software instruction stored within the instruction cache, wherein a functional unit includes a portion of hardware, which is operable to perform a function in response to special instructions received from the processing engine;

identifying unneeded functional units as functional units that are not operable to execute a software instruction stored within the instruction cache;

initiating a power increase for selected ones of the potentially needed functional units that are in a low power state; and

initiating a power decrease for selected ones of the unneeded functional units that are in an operable power state.

16. (Original) The method of claim 15, further comprising:

generating and storing an information vector for selected ones of the one or more lines, wherein the information vector identifies a set of functional units that are operable to execute software instructions within a line; and

wherein identifying the potentially needed functional unit is performed based on the information vector.

17. (Original) The method of claim 15, wherein fetching the one or more instructions comprises fetching the one or more instructions into a conventional cache.

18. (Original) The method of claim 15, wherein fetching the one or more instructions comprises fetching the one or more instructions into a trace cache.

19. (Original) The method of claim 15, wherein initiating the power increase comprises:

initiating the power increase for a functional unit selected from a group of functional units that includes one or more floating-point units, multipliers, dividers, shifters, digital signal processors, co-processors, application specific integrated circuits, data processing engines, debug logic blocks, encryption units and key-generation units.

20. (Original) The method of claim 15, wherein initiating the power increase comprises:

determining a selected operable power level from one of multiple operable power levels, wherein the selected operable power level is selected based on an expected result latency; and

initiating the power increase to the selected operable power level.

21. (Original) A computer-readable medium having program instructions stored thereon to perform a method, which when executed within an electronic system, results in:

identifying a potentially needed functional unit as a functional unit that is operable to execute at least one software instruction stored within an instruction cache; and

initiating a power increase for the potentially needed functional unit, if the potentially needed functional unit has a present power level that is lower than an operable power level.

22. (Original) The computer-readable medium of claim 21, wherein executing the program instructions further results in:

fetching a line of one or more software instructions into the instruction cache;

generating and storing an information vector for the line, wherein the information vector identifies a set of functional units that are operable to execute the one or more software instructions; and

wherein identifying the potentially needed functional unit is performed based on the information vector.

23. (Original) The computer-readable medium of claim 21, wherein executing the program instructions further results in:

selecting one or more selected lines of software instructions stored within the cache; and

wherein identifying the potentially needed functional unit includes identifying the potentially needed functional unit as a functional unit that is operable to execute at least one software instruction stored within the one or more selected lines.

24. (Original) The computer-readable medium of claim 21, wherein executing the program instructions further results in:

identifying an unneeded functional unit as a functional unit that is not operable to execute the at least one software instruction; and

initiating a power decrease for the unneeded functional unit, if the unneeded functional unit has a second present power level that is greater than or equal to a second operable power level.

25. (Original) The computer-readable medium of claim 21, wherein initiating the power increase comprises:

initiating the power increase for a functional unit selected from a group of functional units that includes one or more floating-point units, multipliers, dividers, shifters, digital signal processors, co-processors, application specific integrated circuits, data processing engines, debug logic blocks, encryption units and key-generation units.

26. (Original) An apparatus comprising:

one or more functional units;

an instruction cache;

a processing engine, which is operable to access software instructions stored within the instruction cache, and send one or more special instructions to the one or more functional units in order to execute at least some of the software instructions; and

one or more power controllers, which are operable to control whether or not an operable power level or a low power level is provided to selected ones of the one or more functional units, based on whether or not selected ones of the one or more functional units are operable to execute at least one software instruction stored within the instruction cache.

27. (Original) The apparatus of claim 26, wherein at least one of the one or more functional units includes an internal functional unit, which is located on a same chip as the processing engine.

28. (Original) The apparatus of claim 26, wherein at least one of the one or more functional units includes an external functional unit, which is not located on a same chip as the processing engine.

29. (Original) The apparatus of claim 26, wherein the instruction cache includes a conventional cache.

30. (Original) The apparatus of claim 26, wherein the instruction cache includes a trace cache.

31. (Original) The apparatus of claim 26, wherein the instruction cache comprises:

an array of storage locations; and

a mechanism to sequentially access the storage locations within the array using an enable signal, which has a value that results from shifting information within one or more shift registers.

32. (Original) The apparatus of claim 31, wherein the mechanism to sequentially access the storage locations includes a plurality of first latches, within which a first portion of the enable signal is stored, and wherein the first portion of the enable signal is used to activate a selected word line within the array.

33. (Original) The apparatus of claim 32, wherein the mechanism to sequentially access the storage locations includes a plurality of second latches, within which a second portion of the enable signal is stored, wherein the second portion of the enable signal is used to select a portion of the selected word line.

34. (Original) The apparatus of claim 26, further comprising:

a predecoder, which is operable to evaluate selected ones of the software instructions to determine which functional units will be needed to execute an instruction.

35. (Original) The apparatus of claim 26, further comprising:

a battery interface, operable to provide power to the one or more functional units.

36. (Original) The apparatus of claim 26, further comprising:

a wireless medium interface, operable to enable signals to be sent to and received over a wireless medium.

37. (Original) The apparatus of claim 26, further comprising:

a network interface, operable to enable signals to be sent to and received from a network.

38. (Original) The apparatus of claim 26, wherein the one or more functional units comprise:

one or more functional units selected from a group of functional units that includes one or more floating-point units, multipliers, dividers, shifters, digital signal processors, co-processors, application specific integrated circuits, data processing engines, debug logic blocks, encryption units and key-generation units.

39. (Original) An apparatus comprising:

one or more functional units;

an instruction cache, which includes

an array of storage locations, and

a mechanism to sequentially access the storage locations within the array using an enable signal, which has a value that results from shifting information within one or more shift registers; and

a processing engine, which is operable to access software instructions stored within the instruction cache, and send one or more special instructions to the one or more functional units in order to execute at least some of the software instructions.

40. (Original) The apparatus of claim 39, wherein the mechanism to sequentially access the storage locations includes a plurality of first latches, within which a first portion of the enable signal is stored, and wherein the first portion of the enable signal is used to activate a selected word line within the array.

41. (Original) The apparatus of claim 40, wherein the mechanism to sequentially access the storage locations includes a plurality of second latches, within which a second portion of the enable signal is stored, wherein the second portion of the enable signal is used to select a portion of the selected word line.

42. (Original) The apparatus of claim 39, further comprising:

one or more power controllers, which are operable to control whether or not an operable power level or a low power level is provided to selected ones of the one or more functional units, based on whether or not selected ones of the one or more functional units are operable to execute at least one software instruction stored within the instruction cache.